5-10-2020

Validation of Stereovision Camera System for 3D Strain Tracking

Vaishali Shah
Washington University in St. Louis

Leanne Iannucci
Washington University in St. Louis

Spencer Lake
Washington University in St. Louis

Validation of Stereovision Camera System for 3D Strain Tracking

Vaishali Shah

Introduction:

The ulnar collateral ligament (UCL) is a ligament complex located in the elbow, and it provides stabilization for the joint [1]. Injuries to this tissue are commonly seen in baseball pitchers and other overhand throwing athletes as they perform a repetitive motion that subjects the UCL to cyclic valgus load. These stresses accumulate damage in the UCL that can lead to an ultimate failure event [1]. The Lake Lab is interested in characterizing how region-specific damage accumulates in this tissue during the progressive fatigue loading process. One major necessity in order to complete this analysis is the ability to track tissue deformation or strain across three dimensions. 2D object tracking can be used in these calculations, however there is lost accuracy if the deformation occurs out of the two-dimensional camera plane. The UCL is a curved structure and tissue may be displaced in three dimensions, which means that depth information must be accounted for as well. One way to track objects in 3D and extract depth information is through use of a stereovision camera [2]. These cameras have two lenses that captures simultaneous adjacent views of an object, and then rectifies these images to output a 3D reconstruction of the object (Figure 1).

The stereovision camera used in this project was the commercially available and simple to use FujiFilm FinePix Real 3D W3. Matlab's in-suite computer vision toolbox was used to analyze the images and videos taken by the stereovision camera. In the toolbox, the user may input left and right images taken on the stereovision camera of a calibration frame, which in this case is a

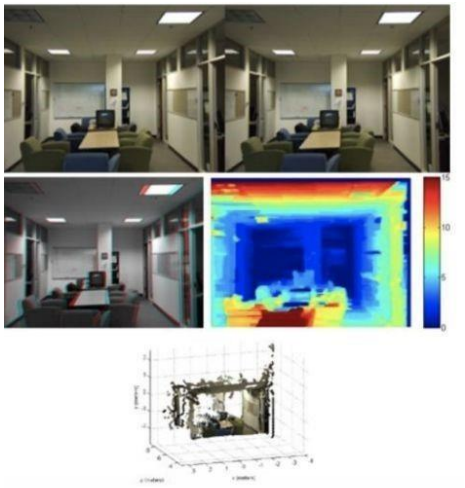

Figure 1. *This image shows the left and right views from two lenses that are combined to create a composite image, image disparity map and a 3D point cloud. Obtained from [3].*

checkerboard of known size. These image pairs will be compared and stereo parameters for the specific setup may be obtained. The stereo parameters contain information on the relationship between the two lenses on the camera and characteristics of the lenses themselves. These parameters can be used for images of test objects in order to gain the 3D coordinates of points in space.

At the end of last semester, a Matlab code had been developed to calibrate the stereovision camera. In this code, a user would choose a file of checkerboard images, input these images into the computer vision toolbox in Matlab, and then the code would output the appropriate stereovision parameters. Using these parameters, the code then calls upon another script to perform the image analysis. This analysis code runs image pairs of a scene of interest through a thresholding interface that is used to separate the objects to be tracked from the background. Once a grayscale is established, the centroids of the points are found and are triangulated into “real world” 3D coordinates using the stereovision parameters and 2D coordinates from each lens view. Then the distance between the objects in three dimensions can be simply calculated.

When running this code several times with a test object, it was observed that there were some points where the test object was placed within the field of view that yielded inaccurate results. When Matlab calculates the stereo parameters, it also outputs a figure showing which frames were used to calibrate the camera and where they were located with respect to the camera lenses. The bounding box that contains the locations of these frames we define as the calibration volume. The error in

measurement was hypothesized to be proportional to the distance the object was from the center of this calibration volume. Thus, the objective for this semester was to measure the distance between points on a test object at different locations in relation to the calibration volume, inside the volume and outside, to determine where the most accurate and inaccurate regions of the camera were. To visualize this relationship, we aimed to create plots of the error as a function of distance from the center of the volume at each zoom setting. These plots will help us determine if we need to calibrate the cameras differently before we move on to experimental testing. These evaluations are essential because his would allow for more accurate measurements during more dynamic testing conditions, like that of the UCL.

Semester Process

Last semester, I conducted a literature review on stereovision imaging and developed Matlab software for implementation of a stereovision system for the Lake Lab. The code took in a matched pair of 2D images from the stereovision camera (FujiFilm FinePix Real 3D W3) and calculated the 3D coordinates of a point in both images. Images were taken on the stereovision camera and split into left and right views by the StereoSplicer application. The code was organized into three files: calibration, analysis/point tracking, and 3D coordinate calculation. The calibration code accepted the input of separated left and right images of checkerboards of a known size. It would input these images into the Matlab stereovision toolbox app [3]. First, the images were evaluated by the software. If one of the checkerboard images in a pair was too blurry or out of the frame of view the software would reject the entire pair. After throwing out the bad images, the program would analyze each of the accepted image pairs by rectifying the left and right images based on the same point on the corners of the checkerboard. Once rectification finished, it went through this same process from each image

pair. Based on these observed locations and the disparity between the points in the image pair, the calibrator calculates the stereovision parameters for the camera. These parameters describe the 3D relationship of the camera lenses to each other in space and allow for depth information to be acquired from two 2D views. Then for each image pair, the program would project a calculated location of the corner point that it found during the rectification process based on the calculated stereovision parameters. Taking the distance between the projected point and the calculated point yields the reprojection error. At the end of the calibration process, a figure showing the image pairs and the mean reprojection error for each pair would be shown and the user could choose to discard those with too high of an error. A high error would be considered to be higher than 1 pixel according to online Matlab documentation [3]. The stereo parameters including the mean reprojection error of the images taken by the user would be updated and then sent to the Matlab workspace to be used in the other two files.

The analysis/point tracking code would accept the files the user inputs from the stereovision camera of left and right views of an area of interest containing points to be tracked. Each view of the scene would be converted to a binary image and a thresholding guided user interface would be used to determine the grayscale level that best distinguishes two points from the background in the photo. After these two points are selected by the user, the centroids of those areas are found by the program as well as the distance between these points. This process then repeats for the other view provided by the adjacent lens of the camera.

After tracking the corresponding 2D coordinates of the points of interest, the third code file would be called. This file performs the triangulation of the 3D coordinates for each point calculation of the points on using the stereo parameters obtained by the previous calibration code. The distance

between these points in 3D can then allow us to calculate the 3D distances between them, and thus calculate 3D tissue deformation and strain.

During this semester, a few issues that I ran into during optimization of this workflow was in the separation of images. Images taken from the stereovision camera have a left and right component from each of the two lenses, but are compiled into one file (MPO for still images and 3D-AVI for videos). Therefore, the left and right views need to be resolved from the filed before they can be uploaded to the program. Last semester, StereoSplicer was used to separate these components. However due to new computer updates (ie. Mac upgrade from High Sierra to Catalina), this app was rendered non usable. The solution that was found was by writing a shell script in bash. The ffmpeg framework was downloaded from homebrew and implemented in a simple script that told the computer what file needed to be split into left and right images. It then saved these split images to a path specified by the user. The advantage of this script is that it should work on any operating system.

After writing this code to separate images, the validation process began. The accuracy of the camera to calculate the distance between two points was evaluated using a phantom test object. This test object was a piece of white paper with two black dots on it with a known distance between them. A few iterations of a workflow were tested in order to get the most efficient validation process. The efficiency was important due to the necessity to repeat this process across every zoom setting on the camera (10 zoom settings). In the end, the initial calibration at a single zoom setting was performed. This involved setting up the stereovision camera to be stationary and moving a checkerboard of known size around the camera at different angles to capture the most frames possible. After running this through the Matlab code two windows were outputted as seen in **Figure**

2. The box showing the x, y and z distances from the camera to the collection of frames was used to create a bounding box (**Figure 3)**. This bounding box was where the calibration app took the frames that it used in the calibration to create the output parameters.

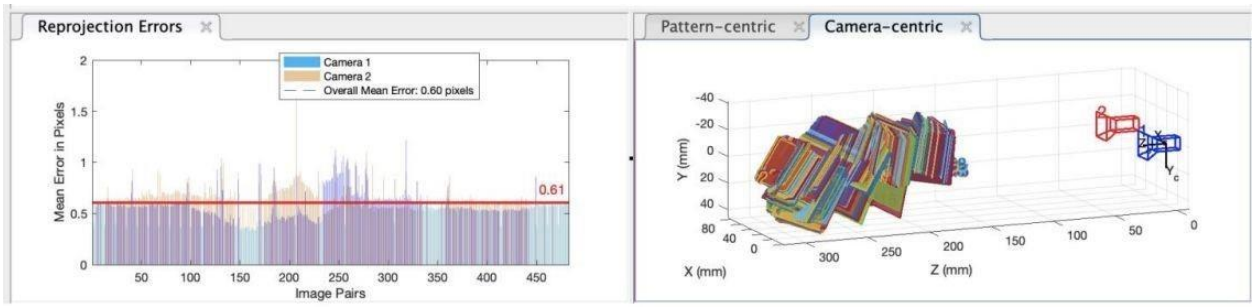

Figure 2. This figure shows the output window during the calibration process. One side shows a graph of the mean reproduction error across all image pairs (left) and one side shows the frames in relation to the position of the lenses spatially (right).

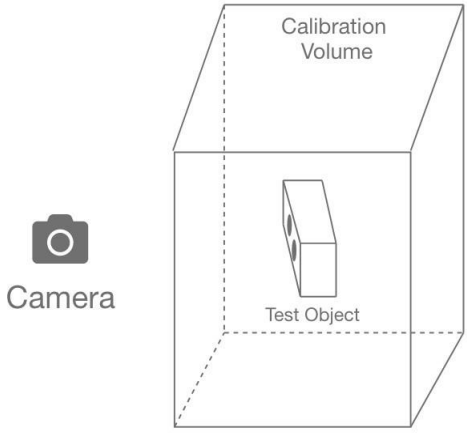

Figure 3. The position of the camera in relation to the bounding box created by the Matlab stereovision program.

To visualize these distances in real life a 2D box made from tape was setup on the lab bench (**Figure 4)**. Using this box, short videos were taken of the object at the given zoom setting. In order to get an accurate estimate of the error, videos were taken within the volume of the box as well as at the edges of the box and far outside of the box volume. This allowed for variety in distance when creating the distance versus error plots.

After a single video of the test object at a location, the location was marked with a piece of tape. Then, the x and y distance from this point to the center of the bounding box were measured. In order to take measurements at points that were not exactly at the z height of the table, a spare box with measured heights written on it was created to measure the z direction. After marking the x, y and z distance from the center of the bounding box as well as the distance between the dots on the test object calculated, an error plot code was writen. After a few iterations of this along with help from Leanne, a software was created that asks the user for the coordinates of the point as well as the distance recorded. This goes through all of the points and plots them on a 3D graph. One thing that is used to show the

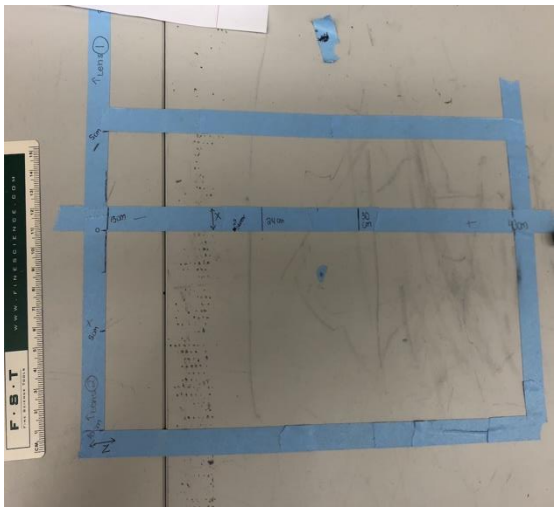

Figure 4. An example of a 2D bounding box created with tape for validation testing.

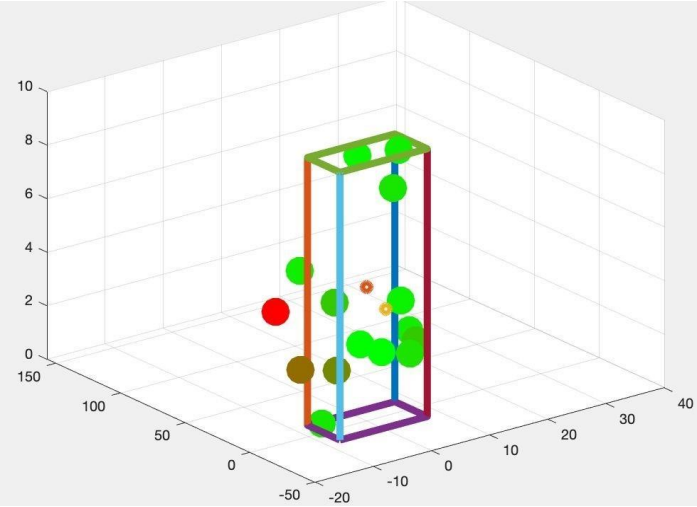

Figure 5. A 3D plot of the bounding box along with colored dots representing the relative error at that specific location. Each dot represents a different sample view for the first zoom setting. The x-axis represents the horizontal distance of the bounding box (-20 to 40 cm) while the y-axis represents the vertical distance of the bounding box (-50 to 150 cm). The z-axis represents the height of the calibration (0 to 10cm). The red dot and the yellow dot represents the center of the box and the camera itself. The red color represents the biggest error for the set of data while the green represents the lowest error for the set of data.

Measurement Number	Percent Error (%)	Location (x,y,z in centimeters)	Color
1	1.1683	(3.67,4.91, 2.25)	
2	5.3728	(6.3, -5.25, 2.25)	
3	1.4539	(3.1, 18.8, 2.25)	
4	22.8778	(-9.31, 9, 2.25)	
5	3.6139	(11.8, 20, 2.25)	
6	3.4111	(4.3, -5.7, 10)	
7	9.0561	(10.15, 7.9, 2.25)	
8	18.5072	(-5.5, -1.7, 2.25)	
9	4.6900	(-5.56, 9.7, 0)	
10	8.9806	(-3.85, 7.2, 4.48)	
11	4.8928	(4.95, 2.05, 8.35)	
12	3.0972	(3.7, -9.65, 4.48)	
13	3.7572	(0, 51.2, 4.48)	
14	38.8161	(-12.5, 14.2, 4.48)	
15	2.2789	(33.37156.6, 4.48)	

Figure 6. These are the measurements from each video taken from this zoom setting. The error for each video, the location that the dot is a in the graph in relation to the bounding box center being (0,0,0) and the corresponding color on the red-green scale are shown here.

relative error between these points is a color scale that matches up with the error calculated. In this case, a more red dot means more error while green means less error. **Figure 5** shows the distance vs error plot for the first zoom setting. The bounding box is shown along with the center of the box. The corresponding errors for each plotted dot in in **Figure 5** can be seen in **Figure 6.**

I continued using this workflow to create these plots for the rest of the zoom settings for the rest of the semester. From this zoom setting plot, the pattern shows that there are more green does within the center of the box. Locations on the sides of the box horizontally, show more error than dots that are further away from the box. This shows that frames that are further away in distance will still be more accurate at this zoom setting that frames that are closer but to the left or right of the camera.

Discussion/Future Work

Overall, this project was focused on validating the code written last semester. Through going through a specific workflow to create plots describing error and location for each zoom setting, our aim was to determine where the camera performs the best (inside the bounding box, at certain distances or angles, etc.) as well as which zoom setting seems to yield the best results. The distance vs error plot that was created for the first zoom setting had mostly low error inside of the bounding box as well as outside of the bounding box at a longer distance. However, there was high error in frames that were taken to the left or right of the bounding box at closer distances. The reason for this error distribution may be because at the positions horizontally, the calibration of the camera did not take into account these locations due to checkerboard blurriness or rejecting by the user due to a high mean error reprojection. Due to these reasons, the camera was never calibrated at these locations,

and will have a high error there. For positions in line with the camera but farther away outside of the bounding box, it could be that for this zoom setting the camera had a very long calibration and is still accurate up to some distance. Due to COVID-19, distance vs error plots could not be created for more zoom settings.

Right before labs moved to alternative operations for the remainder of the spring semester, we had begun exploring alternate ways to calibrate the camera. Although we have established a workflow for camera calibration and validation, the calibration process is time consuming and cumbersome to operate. 100+ image pairs are uploaded into the Matlab stereovision calibrator to perform calibration for one zoom setting and this process overall takes over an hour to conduct. We started looking into other methods of calibration and came across the MultiDIC toolbox [4]. This is an efficient method of calibration as it only requires a single image to calibrate. Current studies are ongoing to evaluate this toolbox and its efficacy to work within our workflow for 3D strain tracking.

The next steps would be to move on to validation testing of the new technique in a more dynamic setting. This dynamic setting would be testing on tissues like bovine tendons before eventually moving onto the ulnar collateral ligament. In these settings, strain can also be calculated. Another goal for this project would be to create a manual detailing the findings of the validation process. It will specify the optimal distances found during the validation process of where the camera should be placed in relation to an object, how many frames should be taken of the object, and where the frames should be taken from. This is so that anyone in the lab can use this software in the future and feel comfortable with the results that it measures.

References:

1. Zaremski, Jason L et al. “Trends in Sports-Related Elbow Ulnar Collateral Ligament Injuries.” *Orthopaedic journal of sports medicine* vol. 5,10 2325967117731296. 16 Oct. 2017, doi:10.1177/2325967117731296

2. Li, Fei-Fei. “Stereo Vision”. 21 October 2014. Standford Vision Lab.

3. “Computer Vision Toolbox.” MATLAB & Simulink, www.mathworks.com/products/computer-vision.html.

4. “MultiDIC: a MATLAB Toolbox for Multi-View 3D Digital Image Correlation” MIT Media Lab, https://www.media.mit.edu/projects/multidic-a-matlab-toolbox-for-multiview-3d-digital-image-correlation/overview/

5. Fusiello, Andrea, et al. “A Compact Algorithm for Rectification of Stereo Pairs.” Machine Vision and Applications, Springer-Verlag, 2 Mar. 2000, http://www.diegm.uniud.it/fusiello/papers/00120016.pdf.